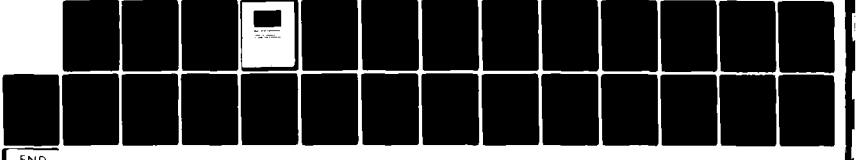
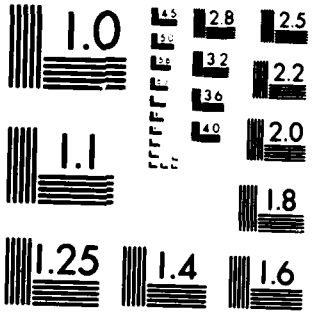
AD-A122 661 LOGARITHMIC DEPTH CIRCUITS FOR ALGEBRAIC FUNCTIONS(U) 1/1
HARVARD UNIV CAMBRIDGE MA AIKEN COMPUTATION LAB J REIF
NOV 82 TR-35-82 N00014-80-C-0674

UNCLASSIFIED F/G 12/1 NL



END
DATE FILMED
DTIC



SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A122661
3. RECIPIENT'S CATALOG NUMBER (2)	
4. TITLE (and Subtitle) Logarithmic Depth Circuits for Algebraic Functions	5. TYPE OF REPORT & PERIOD COVERED Technical Report
	6. PERFORMING ORG. REPORT NUMBER TR-35-82
7. AUTHOR(s) John Reif	8. CONTRACT OR GRANT NUMBER(s) N00014-80-C-0674
9. PERFORMING ORGANIZATION NAME AND ADDRESS Harvard University Cambridge, MA	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research 800 North Quincy Street Arlington, VA 22217	12. REPORT DATE November, 1982
	13. NUMBER OF PAGES 20
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) same as above	15. SECURITY CLASS. (of this report)
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

umlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

polynomials, interpolation, product,division, elementary functions, power series, circuit depth, algebraic computation, evaluation.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

see reverse side

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

82 12

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

This paper describes circuits for computation of various algebraic functions on polynomials, power series, integers, and reals.

Let $\mathcal{R}[x]$ be the polynomials and power series over a commutative ring which supports a fast Fourier transform and let $\mathcal{Q}[x]$ be the polynomials and power series over the rationals $\mathcal{Q}$.

For polynomials of degree n-1, we give circuits of depth O(log n) for computing

- the m-th *power* of a polynomial and the *product* of m polynomials in $\mathcal{R}[x]$, where $m = O(n)$
- the *symmetric functions* on $\mathcal{R}[x]$
- the *remainder* and *quotient* of division of polynomials in $\mathcal{Q}[x]$
- *interpolation* of a polynomial in $\mathcal{Q}[x]$.

For power series with n given low order terms, we give circuits of depth O(log n) for computing the first n low order terms of

- the m-th *power* of a power series in $\mathcal{R}[x]$ and the *product* of m power series in $\mathcal{R}[x]$, where $m = O(n)$
- the *composition* of power series in $\mathcal{R}[x]$
- the *reciprocal* of a power series and the *division* of two power series in $\mathcal{Q}[x]$
- the *reversion* of a power series in $\mathcal{Q}[x]$
- various *elementary functions* applied to power series in $\mathcal{Q}[x]$ such as (fixed) powers, roots, exponentation, logarithm, sin, cos, arctangent, and hyperbolic cosine.

For integers represented by n bit binary numbers, we give boolean circuits (whose gates compute the boolean operations $\wedge$, $\vee$, and $\neg$) of depth $O(\log n(\text{loglog } n)^2)$ for computing:

(Continued on next page)

LOGARITHMIC TIME CIRCUITS
FOR ALGEBRAIC EVALUATION.

John Reif

TR-35-82

Harvard University

Center for Research in Computing Technology

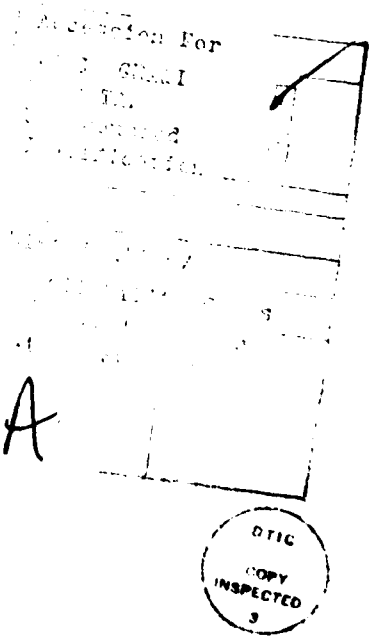
Aiken Computation Laboratory
33 Oxford Street
Cambridge, Massachusetts 02138

LOGARITHMIC DEPTH CIRCUITS FOR ALGEBRAIC FUNCTIONS

John Reif

TR-35-82

November 1982



LOGARITHMIC DEPTH CIRCUITS FOR ALGEBRAIC FUNCTIONS

by

John Reif*

Aiken Computation Laboratory
Division of Applied Science
Harvard University
Cambridge, Massachusetts

*This work was supported in part by the National Science Foundation Grant NSF-MCS79-21024 and the Office of Naval Research Contract N00014-80-C-0647

20. (Continued) Logarithmic Depth Circuits for Algebraic Functions

- the m-th *power* of an integer and the *product* of m integers, where $m = O(n)$
- the *remainder* and *quotient* of the *division* of two integers.

For reals on a finite interval [a,b] represented as floating point numbers within relative accuracy $o(2^{-n})$, we give boolean circuits of depth $O(\log n)(\text{loglog } n)^2)$ for computing within relative accuracy $o(2^{-n})$:

- the m-th *power* of a real and the *product* of m reals where $m = O(n)$
- the *reciprocal* of a real and *division* of reals
- the various *elementary functions* on reals.

As a consequence of the above, for polynomials and power series in $\mathcal{Q}[x]$ we have uniform boolean circuits of depth $O(\log n(\text{loglog } n)^2)$ for all the above listed problems for polynomials and power series, and also:

- *evaluation* of a polynomial or power series in $\mathcal{Q}[x]$ at n points, within relative accuracy $o(2^{-n})$.

All our circuits may be uniformly constructed by a deterministic Turing machine with space $O(\log n)$. The best circuit depth previously known for any of the above problems was $\Omega(\log n)^2$.

0. ABSTRACT

This paper describes circuits for computation of various algebraic functions on polynomials, power series, integers, and reals.

Let $\mathcal{R}[x]$ be the polynomials and power series over a commutative ring which supports a fast Fourier transform and let $\mathcal{Q}[x]$ be the polynomials and power series over the rationals $\mathcal{Q}$.

For polynomials of degree n-1, we give circuits of depth O(log n) for computing

- the m-th *power* of a polynomial and the *product* of m polynomials in $\mathcal{R}[x]$, where $m = O(n)$
- the *symmetric functions* on $\mathcal{R}[x]$
- the *remainder* and *quotient* of division of polynomials in $\mathcal{Q}[x]$
- *interpolation* of a polynomial in $\mathcal{Q}[x]$.

For power series with n given low order terms, we give circuits of depth O(log n) for computing the first n low order terms of

- the m-th *power* of a power series in $\mathcal{R}[x]$ and the *product* of m power series in $\mathcal{R}[x]$ where $m = O(n)$
- the *composition* of power series in $\mathcal{R}[x]$
- the *reciprocal* of a power series and the *division* of two power series in $\mathcal{Q}[x]$
- the *reversion* of a power series in $\mathcal{Q}[x]$
- various *elementary functions* applied to power series in $\mathcal{Q}[x]$ such as (fixed) powers, roots, exponentation, logarithm, sin, cos, arctangent, and hyperbolic cosine.

For integers represented by n bit binary numbers, we give boolean circuits (whose gates compute the boolean operations $\wedge$, $\vee$, and $\neg$) of depth $O(\log n(\text{loglog } n)^2)$ for computing:

- the m-th *power* of an integer and the *product* of m integers, where $m = O(n)$
- the *remainder* and *quotient* of the *division* of two integers.

For reals on a finite interval [a,b] represented as floating point numbers within relative accuracy $o(2^{-n})$, we give boolean circuits of depth $O(\log n)(\text{loglog } n)^2)$ for computing within relative accuracy $o(2^{-n})$:

- the m-th *power* of a real and the *product* of m reals where $m = O(n)$
- the *reciprocal* of a real and *division* of reals
- the various *elementary functions* on reals.

As a consequence of the above, for polynomials and power series in $\mathcal{Q}[x]$ we have uniform boolean circuits of depth $O(\log n(\text{loglog } n)^2)$ for all the above listed problems for polynomials and power series, and also:

- *evaluation* of a polynomial or power series in $\mathcal{Q}[x]$ at n points, within relative accuracy $o(2^{-n})$.

All our circuits may be uniformly constructed by a deterministic Turing machine with space $O(\log n)$. The best circuit depth previously known for any of the above problems was $\Omega(\log n)^2$.

I. INTRODUCTION

Much research is now done on parallel algorithms, although in fact at this time most current computers contain only a single processor. However, most computers do use parallel circuits to implement the most basic and often repeated operations, such as the arithmetic operations: addition, subtraction, multiplication and division. These operations are generally applied to integers with an n bit binary representation, and to floating point reals with relative accuracy 2^{-n}. Other frequently used repeated operations, which certainly would merit special purpose circuits, are the elementary functions such as sin, cosine, arctangent, exponentation, logarithm, square roots, and fixed powers.

The *depth* of a circuit is the time for its parallel execution. What is the minimum depth of boolean circuits for these arithmetic operations and elementary functions?

For integer addition, [Ofman, 62], [Krapchenko, 67] and [Ladner and Fischer, 80] give boolean circuits of depth $O(\log n)$ and size $O(n)$. Subtraction circuits with the same asymptotic depth and size can easily be gotten from these addition circuits.

For integer multiplication, [Ofman, 62] and [Wallace, 64] give boolean circuits of depth $O(\log n)$, and [Schönhage and Strassen, 71] also achieve depth $O(\log n)$ with simultaneous size $O(n(\log n)\operatorname{loglog} n)$.

For division, best known boolean circuit depth was $\Omega(\log n)^2$. [Anderson, et al., 67] first gave such a circuit (which incidentally was implemented by them on the IBM/360 Model 91 Floating-Point Execution Unit). [Knuth, 69] and [Aho, Hopcroft and Ullman, 74] describe a division circuit attributed to Steve Cook of depth $(\log n)^2$ and size $O(n \log n \operatorname{loglog} n)$.

The best known boolean circuit depth for the elementary functions was $\Omega(\log n)^2$ [Brent, 76], [Kung, 76].

Many of the above mentioned boolean circuits of depth $\Omega(\log n)^2$ use a second order Newton iteration with $\Omega(\log n)$ steps, each requiring an n-bit integer multiplication with $\Omega(\log n)$ depth. Alternatively, a reduction is made to the problem of computing the m-th power of a n-bit integer modulo 2^n+1 for $m = O(n)$. This is naively computed by $\Omega(\log n)$ steps of repeated squaring, where each square computation requires $\Omega(\log n)$ depth.

This paper gives a uniform boolean circuits of depth $O(\log n(\operatorname{loglog} n)^2)$ for the problem of computing the product of m n-bit integers modulo (2^n+1). From this result, we get uniform boolean circuits of depth $O(\log n(\operatorname{loglog} n)^2)$ for the problems of division and computing elementary functions, among others.

[Borodin, 77] proved that if a function f is computed in uniform boolean circuit depth $d(n) \geq \log n$, then f can be computed by a deterministic Turing

Machine with space $d(n)$. Thus division and the elementary functions can be computed in deterministic space $O(\log n(\text{loglog } n)^2)$. Note that as an amusing consequence, we have that for any $n \geq 0$ the first n digits of π, Euler's constant e, and the golden ratio φ can all be computed by uniform boolean circuits of depth $O(\log n(\text{loglog } n)^2)$, and hence can be computed in deterministic space $O(\log n(\text{loglog } n)^2)$.

An essential technique in the construction of our product circuit is the use of negatively wrapped convolutions, which can be computed in boolean depth $O(\log n)$ by the fast Fourier transform of [Cooley and Tukey, 65]. This technique was first introduced by [Schönhage and Strassen, 71] for the multiplication of two integers. Our innovation was to generalize the technique to products of more than two integers.

Our techniques are best understood first in the context of polynomials and power series in say $\mathbb{Q}[x]$. In fact, this context is interesting in itself. We might envision a special purpose computer designed for algebraic computation. Its data are (coefficients of) polynomials and power series. The arithmetic operations including division of polynomials and power series are elementary operations of our "algebraic computer." Also, frequently applied operations are the composition of power series, reversion of a power series, computation of elementary functions applied to power series, and interpolation of polynomials.

Section 2 gives circuits of depth $O(\log n)$ that for all these polynomial and power series operations, where each gate of the circuits computes an addition, multiplication, or a division of two rationals. In the case the polynomials and power series have rational coefficients, then we have boolean circuits of $O(\log n(\text{loglog } n)^2)$ depth for all these polynomial and power-series operations. Furthermore, we can also evaluate the resulting polynomials and power series within accuracy $o(2^{-n})$ by boolean circuits with depth $O(\log n(\text{loglog } n)^2)$.

2. CIRCUITS FOR POLYNOMIAL AND POWER SERIES COMPUTATIONS

2.0 Circuit Definitions

A *circuit* α_N over a commutative ring $\mathcal{R} = (\mathcal{D},+,\cdot,0,1)$ is an acyclic labeled digraph, with

(i) a list of N distinguished *input nodes* that have no entering edges

(ii) *constant nodes* with indegree 0 and labeled with constants in $\mathcal{D}$

(iii) *internal nodes* with indegree two and labeled with the symbols in {"+", "·"}

(iv) a list of ℓ distinguished *output nodes*.

Given an assignment of the input nodes from domain $\mathcal{D}$, the value of the circuit at the output nodes is gotten by evaluation of the gates in topological order. The circuit α_N thus defines a mapping from $\mathcal{D}^N$ to $\mathcal{D}^\ell$. A circuit α_N over the rationals $\mathcal{Q}$ is similarly defined, except the nodes can also compute division.

Let f be a function of (the coefficients of) m polynomials $p_1(x),\ldots,p_m(x)$ in $\mathcal{R}[x]$ of degree $n-1$. A circuit α_N for f has $N = mn$ inputs, namely the list of N coefficients in $\mathcal{D}$ of the given polynomials. The output nodes of α_N give the list of coefficients of $f(p_1(x),\ldots,p_m(x))$. If on the other hand f is a function of m power series $p_1(x),\ldots,p_m(x)$ in $\mathcal{R}[x]$ each with n given low order coefficients, then the circuit α_N for f also has $N = nm$ inputs, and the output nodes of α_N only give some prescribed finite number of the coefficients of (the possibly infinite) power series $f(p_1(x),\ldots,p_m(x))$.

The *depth* of circuit α_N is the length of its longest path. A function f over polynomials or power series in R has simultaneous *depth* $O(d(N))$ and size $O(S(N))$ if $\exists$ an infinite family of circuits $\alpha_1,\ldots,\alpha_N,\ldots$ and constants $c_1, c_2 \geq 1$ such that $\forall N \geq 1$, α_N has depth not more than $c_1 d(N)$ and size not more than $c_2 S(N)$ and given N input coefficients of the input polynomial or

power series, α_N computes f within the prescribed number of coefficients.

All the circuits considered in this paper are uniform in the sense of [Borodin, 77]; they may be constructed in space $O(\log N)$ by a deterministic Turing Machine.

2.1 The Discrete Fourier Transform

Fix a commutative ring $\mathcal{R} = (\mathcal{Q}, +, \cdot, 0, 1)$. We assume ω is the principle N-th root of unity in $\mathcal{R}$. Given a vector $a \in \mathcal{R}^N$, the Discrete Fourier Transform is

$$DFT_N(a) = Aa$$

where $A_{ij} = \omega^{ij}$ for $0 \leq i, j < N$. We assume N has a multiplicative inverse and let $A_{ij} = \frac{1}{N}\omega^{-ij}$. The inverse Discrete Fourier Transform is $DFT_N^{-1}(a) = A^{-1}a$ and obviously satisfies $DFT_N^{-1}(DFT_N(a)) = a$. [Cooley and Tukey, 65] gave the Fast Fourier Transform for which

THEOREM 2.1. *DFT_N and DFT_N^{-1} over $\mathcal{R}$ have simultaneous depth $O(\log N)$ and size $O(N \log N)$.*

(Note given a vector $a \in \mathcal{Q}^n$, where $n < N$, $DFT_N(a)$ will be defined to be $DFT_N(a^+)$ where a^+ is the vector of length N derived by concatenating a with N-n zeros.)

2.2 Products of Polynomials

Suppose we are given m vectors $a_i \in \mathcal{Q}^n$ for $i = 1, \ldots, m$. Each vector $a_i = (a_{i,0}, \ldots, a_{i,n-1})^T$ gives the coefficients of a n-1 degree polynomial $A_i(x) = \sum_{j=0}^{n-1} a_{i,j} x^j$ in $\mathcal{R}[x]$. Let $N = nm$. We wish to compute the product polynomial $B(x) = \sum_{k=0}^{N-1} b_k x^k$, where $B(x) = \prod_{i=1}^{m} A_i(x)$. (Note that we have $b_k = 0$ for $N - m + 1 \leq k \leq N - 1$.)

In the special case $m = 2$ and $N = 2n$, the convolution vector $b = (b_0,\ldots,b_{N-1})^T = a_1 \circledast a_2$ gives the coefficients of $B(x)$. By the Convolution Theorem:

$$a_1 \circledast a_2 = DFT_N^{-1}(DFT_N(a_1)\cdot DFT_N(a_2)) \quad \textit{where} \quad \cdot \quad \textit{denotes pairwise product.}$$

Hence the well-known result that

THEOREM 2.2. *The product of two polynomials in* $\mathcal{R}[x]$ *of degree* $n-1$ *has simultaneous depth* $O(\log n)$ *and size* $O(n \log n)$.

In the case of general $m \geq 2$, we wish to compute the coefficient vector

$$b = (b_0,\ldots,b_{N-1})^T = a_1 \circledast \ldots \circledast a_m.$$

By repeated application of the Convolution Theorem, we get

LEMMA 2.1. $b = DFT_N^{-1}(DFT_N(a_1)\cdots DFT_N(a_{N-1}))$.

Thus we first compute in parallel for $i = 1,\ldots,m$ $f_i = DFT_N(a_i)$, where $f_i = (f_{i,0},\ldots,f_{i,N-1})^T$. Next we compute in parallel for $j = 1,\ldots,m$ the elementary products $F_j = \prod_{i=1}^{m} f_{i,j}$. Finally, we compute $DFT((F_0,\ldots,F_{N-1})^T)$. Since the computation of DFT_N, DFT_N^{-1} and the required products F_j, each have depth $O(\log N)$, we have:

THEOREM 2.3. *The product of* m *polynomials in* $\mathcal{R}[x]$ *of degree* $n-1$ *has depth* $O(\log(nm))$.

(Note that the naive method of repeated squaring by Theorem 2.2 has depth $\Omega(\log(m)\log(n))$.

2.3 Modular Products of Polynomials

Let $B(x) = \prod_{i=1}^{n} A_i(x)$ be the product polynomial considered in the previous section. Here we consider the computation of the *modular product* $D(x) = \sum_{i=0}^{n-1} d_i x^i$ where $D(x) \equiv B(x) \bmod (x^n+1)$.

LEMMA 2.2. *The coefficients of* $D(x)$ *are* $d_i = \sum_{r=0}^{m-1} (-1)^r b_{nr+i}$ *for* $i = 0,\ldots,n-1$.

For proof, see the Appendix.

We assume ω is the principle nth root of unity in $\mathcal{R}$, and n has a multiplicative inverse. We also assume there exists an $\psi \in \mathcal{D}$ such that $\psi^2 = \omega$. Then $\psi^n = -1$. Let $\hat{a}_i = (a_{i,0}, \psi a_{i,1}, \ldots, \psi^{n-1} a_{i,n-1})^T$. The *negatively wrapped convolution* of $a_1,\ldots,a_m$ is

$$\hat{d} = (d_0, \psi d_1, \ldots, \psi^{n-1} d_{n-1})^T .$$

In the Appendix we prove:

LEMMA 2.3. $\hat{d} = DFT_n^{-1}(DFT_n(\hat{a}_1)\cdots DFT_n(\hat{a}_m))$.

The above Lemmas 2.2, 2.3 and Theorem 2.1 imply:

THEOREM 2.4. *The modular product* $(A_1(x)\cdots A_m(x)) \bmod (x^n+1)$ *of polynomials* $A_1(x),\cdots,A_m(x)$ *in* $\mathcal{R}[x]$ *degree* $n-1$ *has simultaneous depth* $O(\log(nm))$ *and size* $O(nm \log(nm))$. *The modular power* $A(x)^m \bmod (x^n+1)$ *of a single polynomial* $A(x)$ *of degree* $n-1$ *has simultaneous depth* $O(\log(nm))$ *and size* $O(n \log(nm))$.

2.4 Elementary Functions on Power Series

An immediate consequence of Theorem 2.3 is

COROLLARY 2.1. *The composition of two power series in* $\mathcal{R}[x]$ *has depth* $O(\log n)$.

The elementary functions exp(x), log(x), sin(x), cos(x), arctan(x), and square root(x), etc. all have known Taylor series expansions convergent over given intervals. Thus by Corollary 2.1 we have:

COROLLARY 2.2. *The elementary functions on* $\mathcal{Q}[x]$ *have depth* $O(\log n)$.

For some given $x_1,\ldots,x_N \in \mathcal{D}^N$ it is frequently useful in algebraic computations to determine the polynomial $\prod_{i=1}^{N} (x-x_i) = \sum_{j=0}^{N} (-1)^j p_j x^j$ whose coefficients

$p_j = \Sigma_{i_1<i_2<\cdots<i_j} x_{i_1}\cdots x_{i_j}$ are the elementary symmetric functions. It was pointed out to us by Les Valiant that Theorem 2.3 immediately implies

COROLLARY 2.3. *The elementary symmetric functions in* $\mathcal{R}[x]$ *have depth* $O(\log N)$.

2.5 Power Series and Polynomial Division

Let $A(z) = \sum_{i=0}^{n-1} a_i z^i$ be a power series in $\mathcal{Q}[x]$. The reciprocal of $A(z)$ is the power series $I(z) = \sum_{i=0}^{\infty} r_i z^i$ such that $A(z)\cdot I(z) = 1$. $I(z)$ has the infinite series expansion

$$I(z) = \sum_{i=0}^{\infty} (1-A(z))^i \quad .$$

We wish to compute the first n coefficients of $I(z)$. Since $I(z) = \sum_{i=0}^{n-1} (1-A(z))^i + o(z^n)$, we have by Theorem 2.3:

COROLLARY 2.4. *The first* n *terms of the reciprocal of a power series and the division of two power series in* $\mathcal{Q}[x]$ *can be computed in depth* $O(\log n)$.

An alternative method using the lemma below results in a circuit of depth $O(\log n)$ with smaller circuit size.

LEMMA 2.4. *If* $\tilde{I}(z) = \prod_{i=0}^{\log(n+1)-1} (1-(1-A(z))^{2^i})$ *then* $|I(z) - \tilde{I}(z)| = o(z^n)$ *for* $z \in (0,\frac{1}{2})$ *and* $A(z) > 1 - z$.

For proof, see the Appendix.

In the Appendix we show that Corollary 2.4 implies:

COROLLARY 2.5. *Given polynomials* $a(x)$, $b(x)$ *in* $\mathcal{Q}[x]$ *of degree at most* n, *we can compute in depth* $O(\log n)$ *the unique polynomials* $q(x)$, $r(x)$ *such that* $a(x) = q(x)b(x) + r(x)$ *and degree* $<$ $(r(x))$ $\text{degree}(b(x))$.

2.6 Polynomial Interpolation

COROLLARY 2.6. *Interpolation of a polynomial in* $\mathcal{Q}[x]$ *has depth* $O(\log n)$.

2.7 Reversion of a Power Series

In the Appendix we show that Theorem 2.3 and Corollary 2.4 imply:

COROLLARY 2.7. *The reversion of a power series in* $\mathcal{Q}[x]$ *has depth* $O(\log n)$.

3. INTEGER COMPUTATIONS

3.0 Boolean Circuits

We consider computations over integers given as n bit binary numbers, and reals over $[0,1]$ given within accuracy 2^{-n}. Our computational model in this section is the *boolean circuit*, defined as usual. The i-th input node of α_n takes the i-th bit of the encoding of the input integer or real. Each gate of α_n computes a boolean operation $\vee$, $\wedge$, or $\neg$. Each output node provides a bit of the encoding of the computed integer or real. (In the case of reals with floating point representation, we only provide the input and output bits up to some finite prescribed accuracy.)

3.1 The DFT over an Integer Ring

We assume n and ω are positive powers of two. Let $p = \omega^{n/2} + 1$ and let $\mathcal{R}_p$ be the ring of integers modulo p.

PROPOSITION 3.1. *In* $\mathcal{R}_p$, ω *is the principle* n*th root of unity and* n *has a multiplicative inverse* modulo p.

Proposition 3.1 implies DFT_n and DFT_n^{-1} are well defined.

The fast Fourier transform computation of [Cooley and Tukey, 65] yields a arithmetic circuit α_n of depth $O(\log n)$ and size $O(n \log n)$ computing DFT_n whose elements require:

(i) addition of two '$\log(p)$'-bit integers.

(ii) multiplication of a '$\log(p)$'-bit integer by a power of ω.

We wish to expand α_n into a boolean circuit. Since ω is a power of two, the multiplications can be implemented by the appropriate bit shifts (i.e., the gate connections are shifted by the appropriate amount). The additions can be implemented by Carry-Save Add circuitry of [Ofman, 62] and [Wallice, 64] (also see [Savage, 76]) yielding a boolean circuit of depth $O(\log(np))$ and size $O(np \log(np))$. Thus we have

THEOREM 3.1. *DFT_n and DFT_n^{-1} over integer ring $\mathcal{R}_p$ have simultaneous boolean depth $O(\log(np))$ and size $O(np \log(np))$.*

3.2 Products of Integers

[Schönhage, Strassen, 71] have shown:

THEOREM 3.2. *The product of two N-bit integers has simultaneous boolean depth $O(\log N)$ and size $O(N \log N \operatorname{loglog} N)$.*

We now show:

THEOREM 3.3. *Given a list of N-bit integers $a_1,\ldots,a_m$, the product $(\prod_{i=1}^{m} a_i) \bmod (2^N+1)$ has boolean depth $O(\log(Nm)(\operatorname{loglog} N)^2)$.*

(Note that the naive method of repeated squaring by Theorem 3.2 results in a boolean circuit of depth $\Omega(\log(m) \log N)$.)

Proof. In the case $m > N/(8 \log N)$ we do the computation by partitioning $a_1,\ldots,a_m$ into $\lceil m/N^{1/2} \rceil$ groups, each of size at most $N^{1/2}$. We compute the product of all the elements of each group in parallel by $O(\operatorname{loglog} n)$ iterations of a method described in the proof of Lemma 3.1 below. The result is a list of $\lceil m/N^{1/2} \rceil$ integers of N-bits each.

Our resulting boolean circuit for product will have depth $D(m,N)$. It will satisfy the recurrence

$$D(m,N) = D(\lceil m/N^{1/2} \rceil, N) + D(\lceil N^{1/2} \rceil, N) \quad \text{for } m \geq N/(8 \log N).$$

In the case $m = 1$, we obviously have

$$D(m,N) = 1 .$$

We will prove below:

LEMMA 3.1. *We can construct our boolean circuit for product to satisfy:*

$$D(m,N) = D(m,8\lceil(Nm \log m)^{1/2}\rceil) + O(\log N)$$

for $1 < m < N/(8 \log N)$.

Note that $O(\log\log N)$ applications of the recurrence of Lemma 3.1 implies

$$D(\lceil N^{1/2}\rceil,N) = D(\lceil N^{1/2}\rceil,16\lceil N^{1/2}\log N\rceil) + O(\log N \log\log N) .$$

Solving these above recurrences we get

$$D(m,N) = O(\log(Nm)(\log\log N)^2)$$

for all $m \geq 1$. Thus we have proved Theorem 3.3.

Proof of Lemma 3.1. We can assume we are given N-bit integers $a_1,\ldots,a_m$, where $m < N/(8 \log N)$. We wish to compute $d \equiv b \bmod(2^N+1)$, where $b = \prod_{i=1}^{m} a_i$.

Fix n be the largest power of two not more than $8(Nm \log m)^{1/2}$, and let $\ell = \lceil N/n\rceil$. Each number a_i is subdivided into n "chunks" $a_{i,0},\ldots,a_{i,n-1}$ where $0 \leq a_{i,j} < 2^{\ell}$. Then define the polynomial $A_i(x) = \sum_{j=0}^{n-1} a_{i,j}x^j$ such that $a_i = A_i(2^{\ell})$. The corresponding product polynomial is

$$B(x) = \sum_{i=0}^{nm-1} b_i x^i, \qquad \text{where } B(x) = \prod_{i=1}^{m} A_i(x);$$

it must satisfy $b = B(2^{\ell})$. The modular product polynomial is $D(x) = \sum_{i=0}^{n-1} d_i x^i$, where $D(x) \equiv B(x) \bmod(x^n+1)$; it satisfies $d = D(2^{\ell})$, which is what we have to compute.

In the Appendix we prove:

PROPOSITION 3.2. *For each* $j = 0,\ldots,n-1$, $|d_j| < 2^{2m(\ell+1+\log n)\log m}$.

Let $\omega = 4$ and $p = \omega^{n/2} + 1$. Then by Proposition 3.1, the integer ring $\mathcal{R}_p$ has ω as the principle n-th root of unity and n has a multiplicative inverse mod p. Also, we define $\psi = 2$. Let $\hat{a}_i = (a_{i,0}, \psi a_{i,1}, \ldots, \psi^{n-1} a_{i,n-1})^T$ for $i = 0, \ldots, n-1$. By Lemma 2.2, the coefficients of $D(x)$ are $d_i = \sum_{r=0}^{m-1} (-1)^r b_{nr+i}$ for $i = 0, \ldots, n-1$. By Proposition 3.1, and by our choice of n we have $|d_i| < p/2$ for all $i = 1, \ldots, n-1$. Then $\hat{d} = (d_0, \psi d_1, \ldots, \psi^{n-1} d_{n-1})^T$ is the negatively wrapped convolution of the coefficients of polynomials $A_1(x), \ldots, A_m(x)$. To compute $\hat{d}$, in parallel for $i = 1, \ldots, m$ we compute in the ring $\mathcal{R}_p$, $\mathrm{DFT}(\hat{a}_i) = (g_{i,0}, \ldots, g_{i,n-1})^T$ then in parallel for $k = 0, \ldots, n-1$ we compute $e_k \equiv \prod_{i=0}^{m} g_{i,k} \bmod p$, and finally by Lemma 2.3, $\hat{d} = \mathrm{DFT}_n^{-1}(e_0, \ldots, e_{n-1})$. Since ψ is a power of two, we can easily extract $d_0, \ldots, d_{n-1}$ from $\hat{d}$ in depth $O(\log n)$. By Theorem 3.1, the DFT_n and DFT_n^{-1} computations have depth $O(\log n)$.

Note that since $p = \omega^{n/2} + 1 = 2^n + 1$ and $n < 8(Nm \log m)^{1/2}$, the recurrence claimed in Lemma 3.2 is satisfied. □

3.3 Multiprecision Evaluation of Polynomials and Power Series

Let $p(x)$ be a polynomial or power series in $\mathcal{Q}[x]$ with $n-1$ given rational coefficients of magnitude $< 2^n$. We wish to evaluate $p(x)$ at a floating point real x_0 within relative accuracy $o(2^{-n})$. By Theorem 3.3 we have

COROLLARY 3.1. *The evaluation of* $p(x)$ *at a given* x_0 *to relative accuracy* $o(2^{-n})$ *has boolean depth* $O(\log n (\log\log n)^2)$.

Since the elementary functions exp(x), log(x), sin(x), cos(x), arctan(x), square root(x), etc. power series expansions over given intervals, we have

COROLLARY 3.2. *The evaluation of an elementary function to relative accuracy* $o(2^{-n})$ *has boolean depth* $O(\log n(\log\log n)^2)$.

COROLLARY 3.3. *The elementary symmetric functions (see Section 2.4) over* $\mathbb{Q}[x]$ *have boolean depth* $O(\log n(\log\log n)^2)$.

3.4 Reciprocals and Division of Integers

Let a be an integer within bounds $2^{n-1} \leq a < 2^n$. Then a has binary representation $\sum_{i=0}^{n-1} a_i 2^i$ where $a_{n-1} = 1$. The reciprocal of a is $2^{-(n-1)} r$, where $r = \sum_{i=0}^{\infty} r_i 2^{-i}$. We wish to compute the first n bits $r_0, \ldots, r_{n-1}$. For this, we can use the product form of [Anderson, et al., 67] and [Savage, 76].

LEMMA 3.3. *If* $\tilde{r} = \prod_{i=0}^{\log(n+1)-1} (1-(1-2^{-n}a)^{2^i})$ then $|r-\tilde{r}| = o(2^{-n})$.

By Theorem 3.3 and the above lemma, we get

COROLLARY 3.4. *The reciprocal can be computed within relative accuracy* $o(2^{-n})$ *by a boolean circuit of depth* $O(\log n(\log\log n)^2)$.

COROLLARY 3.5. *Given integers* a, b *with binary representation containing* n *bits, we can compute in boolean depth* $O(\log n(\log\log n)^2)$ *the division quotient* q *and remainder* r *integers such that* $a = qb + r$ *and* $0 \leq r < b$.

Further Results

Our results for $\mathbb{Q}[x]$ can be extended to Euclidean domains. In a forthcoming draft of this paper, we improve the size bounds of our circuitry. Also, we can reduce our boolean depth bound for products in Theorem 3.3 to $O(\log N \log\log N)$ by improving Lemma 3.1 to get the recurrence $D(m, N) = D(m, m'\log N') + O(\log N)$ for $m < N/(8\log N)$.

Acknowledgments

I am grateful to F. Bragdon who first taught me to divide and encouraged me to experiment with faster methods for long divisions.

S. Cook's division circuit stimulated this research. L. Valiant gave some useful criticism of preliminary attempts to develp my division circuit.

REFERENCES

Aho, A.V., Hopcroft, J.E., and Ullman, J.D. (1974). *The Design and Analysis of Computer Algorithms*, Addison Wesley, Reading, Mass.

Anderson, S.F., J.G. Earle, R.E. Goldschmidt, and D.M. Powers, "The IBM System/360 Model 91: Floating Point Multiplication Unit," *IBM J. Research Dev.*, Vol. 11, No. 1, pp. 34-53 (1967).

Borodin, A., J. von zur Gathen, and J. Hopcroft, "Fast Parallel Matrix and GCD Computations," *23rd Annual Symp. on Foundations of Comp. Sci.*, Chicago, Ill., Nov. 1982, pp. 65-71.

Borodin, A., "On Relating Time and Space to Size and Depth," *SIAM J. Comp.*, vol. 6, no. 4, Dec. 1977, pp. 733-744.

Borodin, A. and I. Munro, "The Computation Complexity of Algebraic and Numeric Problems," *American Elsevier*, N.Y., 1975, pp. 77-147.

Brent, R.P., "Fast Multiple-Precision Evaluation of Elementary Functions," *JACM*, vol. 23, no. 2, April 1976, pp. 242-251.

Cooley, J.W. and Tukey, J., "An Algorithm for the Machine Calculation of Complex Fourier Series," *Math. Comp.*, vol. 19, 1965, pp. 297-301.

Hoover, H.J., "Some Topics in Circuit Complexity," Master Thesis, Dept. Comp. Sci., Univ. of Toronto, December 1979.

Knapchenko, A.N., "Asymptotic Estimation of Addition Time of a Parallel Adder," Eng. translation in *Syst. Theory Res.*, vol. 19 (1970), original in *Mat. Zamet.*, vol. 9, no. 1, pp. 35-40.

Knuth, D.E., 1981. *The Art of Computer Programming: Seminumerical Algorithms*, vol. II revised, Addison Wesley, Reading, Mass.

Kung, H.T., "New Algorithms and Lower Bounds for the Parallel Evaluation of Certain Rational Expressions and Recurrences," *JACM*, vol. 23, no. 2, 1976, pp. 252-261.

Ladner, R.E. and M.J. Fischer, "Parallel Prefix Computation," *J. of the Assoc. for Comput. Machinery*, vol. 27, nc. 4, Oct. 1980, pp. 831-838.

Lagrange, *Mémoires Acad. Royale des Sciences et Belles-Lettres de Berlin*,24 (1768), 251-326.

Lipson, J.D., "Chinese Remainder and Interpolation Algorithms," *Proc. of 2nd Symp. on Symbolic and Algebraic Manipulation*, ACM, New York, 1971, pp. 372-391.

Ofman, Y., "On the Algorithmetic Complexity of Discrete Functions," *Sov. Phys. Dokl.*, vol. 7, no. 7, (1963(, pp. 589-591.

Pollard, J., "The Fast Fourier Transform in a Finite Field," *Math. Comp.*, vol. 25, no. 114, pp. 365-374.

Savage, J.E., (1976). *The Complexity of Computing*, John Wiley and Sons, N.Y. pp. 237-260.

Schönhage, A. and Strassen, V., "Schnelle Multiplikation grosser Zahlen," *Computing*, vol. 7, pp. 281-292.

Wallice, C.S., "A Suggestion for a Fast Multiplier," *IEEE Trans. Computing*, vol. EC-13, no. 1, pp. 14-17.

Winograd, S., "On the Time to Perform Multiplication," *JACM*, vol. 14, Oct. 1967, pp. 793-802.

APPENDIX

Proof of Lemma 2.2.

$$B(x) = \sum_{j=0}^{N-1} b_j x^j = \sum_{r=0}^{m-1} \sum_{i=0}^{N-1} b_{nr+i} x^{nr+i}$$

$$\equiv \sum_{r=0}^{m-1} (-1)^r b_{nr+i} x^i \bmod (x^n+1)$$

since $(-1)^r \equiv x^{nr} \bmod (x^n+1)$. □

Proof of Lemma 2.3. For $i = 1,\ldots,m$ let $DFT(\hat{a}_i) = (g_{i,0},\ldots,g_{i,n-1})^T$ where

$$g_{i,k} = \sum_{j=0}^{n-1} a_{i,j} \psi^j \omega^{jk}$$

for $k = 0,\ldots,n-1$. Let

$$e_k = \left(\prod_{i=1}^{m} g_{i,k}\right) = \sum_{0 \leq j_1,\ldots,j_m < n} \psi^{\Sigma j_i} \omega^{k(\Sigma j_i)} \left(\prod_{i=1}^{m} a_{i,j_i}\right).$$

Now let $DFT_n(d) = (e'_0,\ldots,e'_{n-1})^T$. Then for $k = 0,\ldots,n-1$ we let

$$e'_k = \sum_{\ell=0}^{n-1} d_\ell \psi^\ell \omega^{k\ell}$$

$$= \sum_{\ell=0}^{n-1} \sum_{r=0}^{m-1} \psi^\ell \omega^{k\ell} (-1)^r b_{nr+\ell} \qquad \text{by Lemma 2.2}$$

$$= \sum_{\ell=0}^{n-1} \sum_{r=0}^{m-1} \psi^\ell \omega^{k\ell} (-1)^r \sum_{\substack{0 \leq j_1 \ldots j_m < n \\ nr+\ell = \Sigma j_i}} \prod_{i=1}^{m} a_{i,j_i} .$$

But if we substitute $\ell = (\sum_{i=1}^{m} j_i) - nr$ into the above expansion, we get

$$\psi^{\ell}\omega^{k\ell}(-1)^r = \psi^{\sum_i j_i}\omega^{k(\sum_i j_i)}$$

since $\psi^{nr} = (-1)^r$ and $\omega^n = 1$. Hence $e'_k = e_k$. □

Proof of Lemma 2.4. Let $B(z) = 1 - A(z)$. Then $A(z)\tilde{I}(z) = (1-B(z))\tilde{I}(z) = 1 - B(z)^{n+1} = 1 - (1-A(z))^{n+1}$. So

$$\begin{aligned} |I(z) - \tilde{I}(z)| &= \frac{(1-A(z))^{n+1}}{A(z)} \\ &\leq 2(1-A(z))^{n+1} && \text{since} \quad A(z) \geq \tfrac{1}{2} \\ &\leq 2z^{n+1} && \text{since} \quad z \geq 1 - A(z) \\ &= o(z^n) && \text{since} \quad z \in (0, \tfrac{1}{2}) . \end{aligned}$$ □

Proof of Corollary 2.5. (Also, see [Knuth, 81]). Let $n_1 = \text{degree}(a(x))$ and $n_2 = \text{degree}(b(x))$. The computation is trivial unless $n_1 \geq n_2 \geq 1$. Then

$$A(z) = Q(z)B(z) + z^{n_1-n_2+1}R(z)$$

where

$$A(z) = z^{n_1}a(\tfrac{1}{z}), \quad B(z) = z^{n_2}b(\tfrac{1}{z}), \quad Q(z) = z^{n_1-n_2}q(\tfrac{1}{z})$$

and $$R(z) = z^{n_2-1}r(\tfrac{1}{z}).$$

Thus to compute the coefficients of $q(x)$, $r(x)$ we compute the first $n_2 - n_1 + 1$ coefficients of $A(z)/B(z) = Q(z) + O(z^{n_1-n_2+1})$, then compute the power series $A(z) - B(z)Q(z) = z^{n_1-n_2+1}R(z)$, and finally output the coefficients of $Q(z)$, $R(z)$. □

Proof of Corollary 2.6. Suppose we are given $p_1(x),\ldots,p_m(x)$ polynomials in $\mathcal{Q}[x]$ each of degree n-1, and polynomials $q_1(x),\ldots,q_m(x)$ where

degree$(q_i(x)) <$ degree$(p_i(x))$ for $i = 1,\ldots,n$. Let $P(x) = \prod_{i=1}^{m} p_i(x)$. The Chinese Remainder Theorem states that there is a unique polynomial $Q(x)$ of degree less than that of $P(x)$ such that $Q(x) \equiv q_i(x) \bmod p_i(x)$ for $i = 1,\ldots,m$.

The Lagrangian interpolation formula gives

$$Q(x) \equiv \sum_{i=0}^{m} q_i(x) r_i(x) s_i(x) \bmod P(x)$$

where $s_i(x) = P(x)/p_i(x)$ and $r_i(x)$ is the multiplicative inverse of $s_i(x) \bmod p_i(x)$.

Theorem 2.2 and Corollary 2.5 imply that preconditioned Chinese remaindering, with the $r_1(x),\ldots,r_m(x)$ also given, has depth $O(\log n)$.

However, in the special case $p_i(x) = x - a_i$ for $i = 1,\ldots,m$, where the a_i are distinct then each $r_i(x) = 1/s_i(x)$ can be computed in parallel by Theorem 3.3 and Corollary 2.5 in depth $O(\log n)$. In this case the $q_i(x) = b_i$ are constants, since they must have degree less than the $p_i(x)$.

Further note that in this case $Q(x)$ is the unique polynomial such that $Q(a_i) = b_i$ for $i = 1,\ldots,m$. Thus we have proved Corollary 2.6. □

Proof of Corollary 2.7. Let $A(x) = \sum_{i=0}^{\infty} a_i x^i$ be a power series in $Q[x]$ where $a_0 = 0$ and $a_1 = 1$. The *reversion* of $A(x)$ is the power series $R(z) = \sum_{k=0}^{\infty} r_k z^k$ where $z = A(x)$. Note that $r_0 = 0$ and $r_1 = 1$. For the kth coefficient, we first compute

$$B(x) = \frac{1}{A(x)^k} = \sum_{i=0}^{\infty} b_i x^i ,$$

and then apply Lagrange's reversion formula [Lagrange, 1768] $r_k = b_{k-1}/k$ for $k \geq 2$. Thus Theorem 3.3 implies Corollary 2.7. □

Proof of Proposition 3.2. Let $f(i)$ be the maximum magnitude of any coefficient of a polynomial resulting from a product of 2^i of the $A_j(x)$ polynomials taken $\bmod(x^n+1)$. Clearly $f(C) = 2^{\ell}$ and $f(i) = 2n\ f(i-1)^2$ for $i > 0$. Solving this recurrence we get

$$f(\lceil \log m \rceil) < 2^{2m(\ell+1+\log n)\log m}.$$ □